1. 上课约定须知

课程主题：全域离线数仓 -- 第三次 -- PB级全域电商数仓实践
上课时间：20:00 - 23:00
课件休息：21:30 左右 休息10分钟
课前签到：如果能听见音乐，能看到画面，请在直播间扣 666 签到

2. 上次作业复盘

2.1. 数据格式

原始数据：access.log

样例：

```
58.215.204.118 - - [18/Sep/2013:06:51:35 +0000] "GET /wp-includes/js/jquery/jquery.js?ver=1.10.2 HTTP/1.1" 304 0
"http://blog.fens.me/nodejs-socketiochat/" "Mozilla/5.0 (Windows NT 5.1; rv:23.0) Gecko/20100101 Firefox/23.0"
```

字段解释：

```
1、访客 ip 地址： 58.215.204.118
2、访客用户信息： - -
3、请求时间： [18/Sep/2013:06:51:35 +0000]
4、请求方式： GET
5、请求的 url： /wp-includes/js/jquery/jquery.js?ver=1.10.2
6、请求所用协议： HTTP/1.1
7、响应码： 304
8、返回的数据流量： 0
9、访客的来源 url： http://blog.fens.me/nodejs-socketio-chat/
10、访客所用浏览器： Mozilla/5.0 (Windows NT 5.1; rv:23.0) Gecko/20100101 Firefox/23.0
```

2.2. 需求实现

1、清洗上述数据，得到规整的结果：

时间戳	IP地址	请求URL	Referal	浏览器属性	……
2012-01-01 12:31:12	101.0.0.1	/a/…	somesite.com		
2012-01-01 12:31:16	201.0.0.2	/a/…	aura.cn		
2012-01-01 12:33:06	101.0.0.2	/b/…	baidu.com		
2012-01-01 15:16:39	234.0.0.3	/c/…	google.com		
2012-01-01 15:17:11	101.0.0.1	/d/…	/c/…		
2012-01-01 15:19:23	101.0.0.1	/e/…	/d/….		

2、从上述清洗结果中梳理出以下两种模型数据

第一种：页面点击流模型 Pageviews 表

SessionID	userid	时间	访问页面URL	页面停留时长	第几步
S001	User01	2012-01-01 12:31:12	/a/….	30	1
S002	User02	2012-01-01 12:31:16	/a/….	110	1
S002	User02	2012-01-01 12:33:06	/b/….	120	2
S002	User02	2012-01-01 12:35:06	/e/….	30	3

第二种：点击流模型 Visits 表

Session	起始时间	结束时间	进入页面	离开页面	访问页面数	IP	cookie	referal
S001	2012-01-01 12:31:12	2012-01-01 12:31:12	/a/…	/a/…	1	101.0.0.1	User01	somesite.com
S002	2012-01-01 12:31:16	2012-01-01 12:35:36	/a/…	/e/…	3	201.0.0.2	User02	naixue.cn
S003	2012-01-01 12:35:42	2012-01-01 12:35:42	/c/…	/c/…	1	234.0.0.3	User03	baidu.com
S004	2012-01-01 15:16:39	2012-01-01 15:19:23	/c/…	/e/…	3	101.0.0.1	User01	google.com
……	……	……	……	……	……	……	……	……

3. 上次课程复习

上次课程是《全域离线数仓》的 第二次课程，课程的主要内容有：

```
1、构建全域数仓背景与目标
2、全域数仓需求分析：痛点和解决方案
3、PB级全域数仓构建：各种规范
4、PB级数据采集平台架构设计
5、PB级数仓平台资源评估
6、数据治理功能图鉴
```

主要讲解的是一些关于数仓构建的一些规范。最终，都是要形成一个 约束： 开发一个 web UI 平台

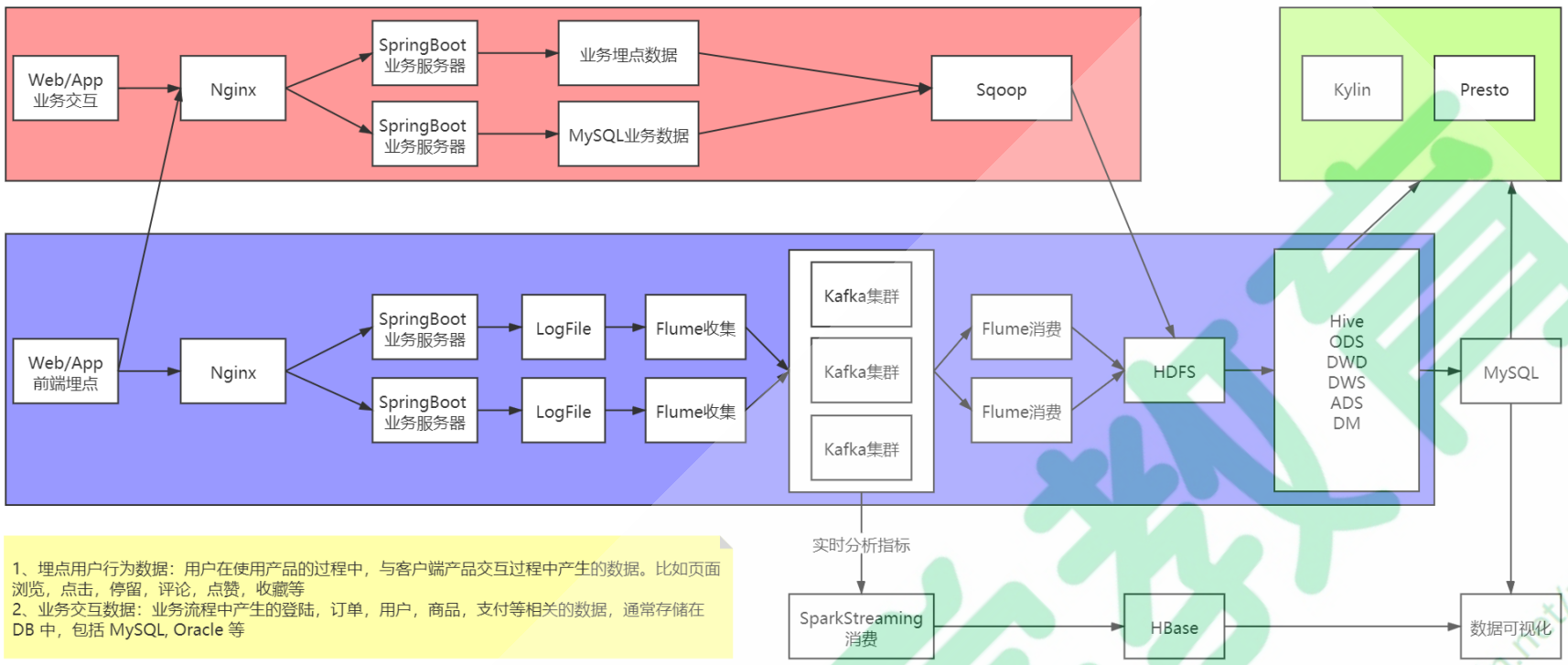
4. 本次内容预告

本次课程是《全域离线数仓》的 第三次课程，主要讲解大厂企业实战：

1、离线数仓 ODS 层建设
2、离线数仓 DWD 层建设
3、离线数仓 DWS 层建设
4、离线数仓 ADS 层应用统计分析大厂企业案例
5、订单拉链表设计

5. 全域 PB 级数仓实践

5.1. 数仓搭建ODS、DWD与DWS层



5.1.1. 关于数据

准备了两份资料：

```
1、数据
    两种类型的数据
        1、业务库的数据：订单，商品，用户，等
        2、前端埋点收集到的日志数据：app启动日志，用户行为日志
2、脚本
    ODS，DWD,DWS 层的数据预处理的 shell 脚本
```

5.1.2. 创建数据库

1、创建数据库的相关命令：

```
create database if not exists hdp_zhuanzhuan_ods_global comment "ODS层原始数据";
create database if not exists hdp_zhuanzhuan_dwd_global comment "DWD层明细数据";
create database if not exists hdp_zhuanzhuan_dws_global comment "DWS轻度汇总层";
create database if not exists hdp_zhuanzhuan_ads_global comment "ADS数据应用层";
create database if not exists hdp_zhuanzhuan_dim_global comment "DIM公共维度层";
create database if not exists hdp_zhuanzhuan_tmp_global comment "TMP临时数据层";
```

说明：如果数据库存在且有数据，需要强制删除时执行：drop database xxx cascade;

2、使用数据库(举例：hdp_zhuanzhuan_ods_global)

```
use hdp_zhuanzhuan_ods_global;
```

5.1.3. 创建ODS层表

ODS 层有 10 张表：

```
1、第一大类：业务数据库的 8 张表
2、第二大类：用户行为日志表 2 张表
```

5.1.3.1. 运营数据库表-清单

序号	运营数据库表名	表中文名	ods层表名	表同步策略
1	zz_order_info	订单表	ods_zz_order_info_inc_1d	新增及变化
2	zz_order_detail	订单详情表	ods_zz_order_detail_inc_1d	增量
3	zz_payment_info	支付流水表	ods_zz_payment_info_inc_1d	增量
4	zz_sku_info	商品表	ods_zz_sku_info_full_1d	全量
5	zz_user_info	用户表	ods_zz_user_info_full_1d	全量
6	zz_category1	商品一级分类表	ods_zz_category1_full_1d	全量
7	zz_category2	商品二级分类表	ods_zz_category2_full_1d	全量
8	zz_category3	商品三级分类表	ods_zz_category3_full_1d	全量

5.1.3.2. 创建表

完全仿照业务数据库中的表字段，一模一样的创建 ODS 层对应表。

订单表

```
use hdp_zhuanzhuan_ods_global;
drop table if exists ods_zz_order_info_inc_1d;
create EXTERNAL table ods_zz_order_info_inc_1d (
    `id` string COMMENT '订单编号',
    `total_amount` decimal(10,2) COMMENT '订单金额',
    `order_status` string COMMENT '订单状态',
    `user_id` string COMMENT '用户id' ,
    `payment_way` string COMMENT '支付方式',
    `out_trade_no` string COMMENT '支付流水号',
    `create_time` string COMMENT '创建时间',
    `operate_time` string COMMENT '操作时间'
) COMMENT '订单表'
PARTITIONED BY (`dt` string)
row format delimited  fields terminated by '\t';
```

订单详情表

```
use hdp_zhuanzhuan_ods_global;
drop table if exists ods_zz_order_detail_inc_1d;
create EXTERNAL table ods_zz_order_detail_inc_1d(
    `id` string COMMENT '订单编号',
    `order_id` string  COMMENT '订单号',
    `user_id` string COMMENT '用户id' ,
    `sku_id` string COMMENT '商品id',
    `sku_name` string COMMENT '商品名称',
    `order_price` string COMMENT '下单价格',
    `sku_num` string COMMENT '商品数量',
    `create_time` string COMMENT '创建时间'
) COMMENT '订单明细表'
PARTITIONED BY ( `dt` string)
row format delimited  fields terminated by '\t';
```

支付流水表

```
use hdp_zhuanzhuan_ods_global;
drop table if exists ods_zz_payment_info_inc_1d;
create EXTERNAL table ods_zz_payment_info_inc_1d(
    `id`              bigint COMMENT '编号',
    `out_trade_no`    string COMMENT '对外业务编号',
    `order_id`        string COMMENT '订单编号',
    `user_id`         string COMMENT '用户编号',
    `alipay_trade_no` string COMMENT '支付宝交易流水编号',
    `total_amount`    decimal(16,2) COMMENT '支付金额',
    `subject`         string COMMENT '交易内容',
    `payment_type`    string COMMENT '支付类型',
    `payment_time`    string COMMENT '支付时间'
) COMMENT '支付流水表'
PARTITIONED BY ( `dt` string)
row format delimited  fields terminated by '\t';
```

商品表

```
use hdp_zhuanzhuan_ods_global;
drop table if exists ods_zz_sku_info_full_1d;
```

```
create EXTERNAL table ods_zz_sku_info_full_1d(
    `id` string COMMENT 'skuId',
    `spu_id` string   COMMENT 'spuid',
    `price` decimal(10,2) COMMENT '价格' ,
    `sku_name` string COMMENT '商品名称',
    `sku_desc` string COMMENT '商品描述',
    `weight` string COMMENT '重量',
    `tm_id` string COMMENT '品牌id',
    `category3_id` string COMMENT '品类id',
    `create_time` string COMMENT '创建时间'
) COMMENT '商品表'
PARTITIONED BY ( `dt` string)
row format delimited  fields terminated by '\t';
```

用户表

```
use hdp_zhuanzhuan_ods_global;
drop table if exists ods_zz_user_info_full_1d;
create EXTERNAL table ods_zz_user_info_full_1d(
    `id` string COMMENT '用户id',
    `name`  string COMMENT '姓名',
    `birthday` string COMMENT '生日',
    `gender` string COMMENT '性别',
    `email` string COMMENT '邮箱',
    `user_level` string COMMENT '用户等级',
    `create_time` string COMMENT '创建时间'
) COMMENT '用户信息'
PARTITIONED BY ( `dt` string)
row format delimited  fields terminated by '\t';
```

商品一级分类表

```
use hdp_zhuanzhuan_ods_global;
drop table if exists ods_zz_category1_full_1d;
create EXTERNAL table ods_zz_category1_full_1d(
    `id` string COMMENT 'id',
    `name`  string COMMENT '名称'
) COMMENT '商品一级分类'
PARTITIONED BY ( `dt` string)
row format delimited  fields terminated by '\t';
```

商品二级分类表

```
use hdp_zhuanzhuan_ods_global;
drop table if exists ods_zz_category2_full_1d;
create EXTERNAL table ods_zz_category2_full_1d(
    `id` string COMMENT ' id',
    `name`  string COMMENT '名称',
    category1_id string COMMENT '一级品类id'
) COMMENT '商品二级分类'
PARTITIONED BY (`dt` string)
row format delimited  fields terminated by '\t';
```

商品三级分类表

```
use hdp_zhuanzhuan_ods_global;
drop table if exists ods_zz_category3_full_1d;
create EXTERNAL table ods_zz_category3_full_1d(
    `id` string COMMENT ' id',
    `name`  string COMMENT '名称',
    category2_id string COMMENT '二级品类id'
) COMMENT '商品三级分类'
PARTITIONED BY (`dt` string)
row format delimited  fields terminated by '\t';
```

5.1.3.3. 行为埋点日志-清单

序号	埋点日志类型	表中文名	ods层表名	更新方式
1	用户启动日志	启动日志表	ods_log_start_app_inc_1d	增量
2	用户行为日志	事件日志表	ods_log_event_inc_1d	增量

原始数据层，存放原始数据，直接加载原始日志、数据，数据保持原貌不做处理。

5.1.3.4. 创建表

启动日志表

```
use hdp_zhuanzhuan_ods_global;
drop table if exists ods_log_start_app_inc_1d;
create EXTERNAL table ods_log_start_app_inc_1d(
    `line`  string COMMENT '日志行'
) COMMENT '启动日志表'
PARTITIONED BY (`dt` string)
row format delimited  fields terminated by '\t';
```

事件日志表

```
use hdp_zhuanzhuan_ods_global;
drop table if exists ods_log_event_inc_1d;
create EXTERNAL table ods_log_event_inc_1d(
    `line`  string COMMENT '日志行'
) COMMENT '事件日志表'
PARTITIONED BY (`dt` string)
row format delimited  fields terminated by '\t';
```

关于这 10 张表还有 2 个细节：

```
1、表的数据存储格式：ods层，textfile，补充一点：dwd dws parquet+snappy
2、所有的表都是 external 表，主要目的是防止数据删除
```

5.1.3.5. 准备数据

将 数据文件 和 脚本文件 存放在 Linux 本地

```
put -r origin_data
put -r warehouse
```

origin_data 数据结构：

```
origin_data/
├── db
|   ├── zz_category1
|   |   ├── 2020-09-18
|   |   |   └── zz_category1.txt
|   |   └── 2020-09-19
|   |       └── zz_category1.txt
|   ├── zz_category2
|   |   ├── 2020-09-18
|   |   |   └── zz_category2.txt
|   |   └── 2020-09-19
|   |       └── zz_category2.txt
|   ├── zz_category3
|   |   ├── 2020-09-18
|   |   |   └── zz_category3.txt
|   |   └── 2020-09-19
|   |       └── zz_category3.txt
|   ├── zz_order_detail
|   |   ├── 2020-09-18
|   |   |   └── order_detail-2020-09-18.txt
|   |   └── 2020-09-19
|   |       └── order_detail-2020-09-19.txt
|   ├── zz_order_info
|   |   ├── 2020-09-18
|   |   |   └── order_info-2020-09-18.txt
|   |   └── 2020-09-19
|   |       └── order_info-2020-09-19.txt
|   ├── zz_payment_info
|   |   ├── 2020-09-18
|   |   |   └── payment_info-2020-09-18.txt
|   |   └── 2020-09-19
|   |       └── payment_info-2020-09-19.txt
|   ├── zz_sku_info
|   |   ├── 2020-09-18
|   |   |   └── sku-info-2020-09-18.txt
|   |   └── 2020-09-19
|   |       └── sku-info-2020-09-19.txt
|   └── zz_user_info
|       ├── 2020-09-18
|       |   └── user_info-2020-09-18.txt
|       └── 2020-09-19
|           └── user_info-2020-09-19.txt
└── log
```

```
    ├── event
    |   ├── 2020-09-18
    |   |   └── event-2020-09-18.txt
    |   └── 2020-09-19
    |       └── event-2020-09-19.txt
    └── start_app
        ├── 2020-09-18
        |   └── start_app_2020-09-18.txt
        └── 2020-09-19
            └── start_app-2020-09-19.txt
```

warehouse 数据结构：

```
warehouse/
├── dwd
|   └── bin
|       └── etl_dwd_data.sh
├── dws
|   └── bin
|       └── etl_dws_data.sh
└── ods
    └── bin
        ├── load_data1.sh
        └── load_data.sh
```

5.1.3.6. 脚本导入ODS层数据

1、在 /home/bigdata/warehouse/ods/bin 下面创建 load_data.sh

```
cd /home/bigdata/warehouse/ods/bin/
vim /home/bigdata/warehouse/ods/bin/load_data.sh
```

在脚本中填写如下内容

```
#!/bin/bash

do_date=$1
dbname=hdp_zhuanzhuan_ods_global
hive=/home/bigdata/apps/apache-hive-3.1.2-bin/bin/hive

sql="
load data local inpath '/home/bigdata/origin_data/db/zz_order_info/$do_date'  OVERWRITE into table
$dbname"".ods_zz_order_info_inc_1d partition(dt='$do_date');

load data local inpath '/home/bigdata/origin_data/db/zz_order_detail/$do_date'  OVERWRITE into table
$dbname"".ods_zz_order_detail_inc_1d partition(dt='$do_date');

load data local inpath '/home/bigdata/origin_data/db/zz_payment_info/$do_date'  OVERWRITE into table
$dbname"".ods_zz_payment_info_inc_1d partition(dt='$do_date');

load data local inpath '/home/bigdata/origin_data/db/zz_sku_info/$do_date'  OVERWRITE into table
$dbname"".ods_zz_sku_info_full_1d partition(dt='$do_date');

load data local inpath '/home/bigdata/origin_data/db/zz_user_info/$do_date' OVERWRITE into table
$dbname"".ods_zz_user_info_full_1d partition(dt='$do_date');

load data local inpath '/home/bigdata/origin_data/db/zz_category1/$do_date' OVERWRITE into table
$dbname"".ods_zz_category1_full_1d partition(dt='$do_date');

load data local inpath '/home/bigdata/origin_data/db/zz_category2/$do_date' OVERWRITE into table
$dbname"".ods_zz_category2_full_1d partition(dt='$do_date');

load data local inpath '/home/bigdata/origin_data/db/zz_category3/$do_date' OVERWRITE into table
$dbname"".ods_zz_category3_full_1d partition(dt='$do_date');

load data local inpath '/home/bigdata/origin_data/log/start_app/$do_date' OVERWRITE into table
$dbname"".ods_log_start_app_inc_1d partition(dt='$do_date');

load data local inpath '/home/bigdata/origin_data/log/event/$do_date' OVERWRITE into table
$dbname"".ods_log_event_inc_1d partition(dt='$do_date');
"
$hive -e "$sql"
```

2、添加脚本执行权限

```
chmod 777 /home/bigdata/warehouse/ods/bin/load_data.sh
```

3、执行脚本导入数据

```
cd /home/bigdata/warehouse/ods/bin/
sh load_data.sh 2020-09-18
sh load_data.sh 2020-09-19
```

4、检测数据

```
select * from ods_zz_user_info_full_1d limit 3;
select * from ods_zz_sku_info_full_1d limit 3;
select * from ods_zz_order_info_inc_1d limit 3;
select * from ods_zz_order_detail_inc_1d limit 3;
select * from ods_zz_payment_info_inc_1d limit 3;
select * from ods_zz_category1_full_1d limit 3;
select * from ods_zz_category2_full_1d limit 3;
select * from ods_zz_category3_full_1d limit 3;
select * from ods_log_start_app_inc_1d limit 3;
select * from ods_log_event_inc_1d limit 3;
```

```
Hive 检查:
desc formatted hdp_zhuanzhuan_ods_global.ods_zz_sku_info_full_1d;

HDFS 检查:
hadoop fs -ls hdfs://hadoop330ha/user/hive/warehouse312/hdp_zhuanzhuan_ods_global.db/ods_zz_sku_info_full_1d
```

5、检查数据：三张商品分类表的join查询

```
set hive.cli.print.header=true;
use hdp_zhuanzhuan_ods_global;
select
c1.id,
c1.name,
c2.id,
c2.name,
c3.id,
c3.name
from ods_zz_category3_full_1d c3
join ods_zz_category2_full_1d c2 on (c3.category2_id=c2.id)
join ods_zz_category1_full_1d c1 on (c2.category1_id=c1.id)
where c1.dt='2020-09-18' and c2.dt='2020-09-18' and c3.dt='2020-09-18'
limit 100;
```

如果为了提供效率，可以考虑使用本地化优化：

```
set hive.exec.mode.local.auto=true;
```

10张表：

```
1、8 张来自于 MySQL 业务数据库的 两天数据的 表
2、2 张来自于 前端埋点收集到的用户行为数据 表
```

5.1.4. DWD层数据解析

对 ODS 层数据进行清洗（去除空值，脏数据，超过极限范围的数据，行式存储改为列存储，改压缩格式，维度退化，脱敏等），具体还需看自己的数据情况。

5.1.4.1. 创建DWD表

创建订单表

```
use hdp_zhuanzhuan_dwd_global;
drop table if exists dwd_mysql_order_info_inc_1d;
create external table dwd_mysql_order_info_inc_1d (
    `id` string COMMENT '订单编号',
    `total_amount` decimal(10,2) COMMENT '订单金额',
    `order_status` string COMMENT '订单状态',
    `user_id` string COMMENT '用户id' ,
    `payment_way` string COMMENT '支付方式',
    `out_trade_no` string COMMENT '支付流水号',
    `create_time` string COMMENT '创建时间',
```

```
    `operate_time` string COMMENT '操作时间'
) COMMENT '订单表'
PARTITIONED BY (`dt` string)
stored as parquet
tblproperties ("parquet.compression"="snappy");
```

创建订单详情表

```
use hdp_zhuanzhuan_dwd_global;
drop table if exists dwd_mysql_order_detail_inc_1d;
create EXTERNAL table dwd_mysql_order_detail_inc_1d(
    `id` string COMMENT '订单编号',
    `order_id` string  COMMENT '订单号',
    `user_id` string COMMENT '用户id' ,
    `sku_id` string COMMENT '商品id',
    `sku_name` string COMMENT '商品名称',
    `order_price` string COMMENT '下单价格',
    `sku_num` string COMMENT '商品数量',
    `create_time` string COMMENT '创建时间'
) COMMENT '订单明细表'
PARTITIONED BY ( `dt` string)
stored as  parquet
tblproperties ("parquet.compression"="snappy");
```

创建支付流水表

```
use hdp_zhuanzhuan_dwd_global;
drop table if exists dwd_mysql_payment_info_inc_1d;
create EXTERNAL table dwd_mysql_payment_info_inc_1d(
     `id`   bigint COMMENT '编号',
    `out_trade_no`    string COMMENT '对外业务编号',
    `order_id`        string COMMENT '订单编号',
    `user_id`         string COMMENT '用户编号',
    `alipay_trade_no` string COMMENT '支付宝交易流水编号',
    `total_amount`    decimal(16,2) COMMENT '支付金额',
    `subject`         string COMMENT '交易内容',
    `payment_type` string COMMENT '支付类型',
    `payment_time`   string COMMENT '支付时间'
) COMMENT '支付流水表'
PARTITIONED BY ( `dt` string)
stored as  parquet
tblproperties ("parquet.compression"="snappy");
```

创建用户表

```
use hdp_zhuanzhuan_dwd_global;
drop table if exists dwd_mysql_user_info_full_1d;
create EXTERNAL table dwd_mysql_user_info_full_1d(
    `id` string COMMENT '用户id',
    `name`  string COMMENT '姓名',
    `birthday` string COMMENT '生日',
    `gender` string COMMENT '性别',
    `email` string COMMENT '邮箱',
    `user_level` string COMMENT '用户等级',
    `create_time` string COMMENT '创建时间'
) COMMENT '用户信息'
PARTITIONED BY ( `dt` string)
stored as  parquet
tblproperties ("parquet.compression"="snappy");
```

创建商品表（加分类，降维）

```
use hdp_zhuanzhuan_dwd_global;
drop table if exists dwd_mysql_sku_info_full_1d;
create external table dwd_mysql_sku_info_full_1d(
    `id` string COMMENT 'skuId',
    `spu_id` string   COMMENT 'spuid',
    `price` decimal(10,2) COMMENT '价格',
    `sku_name` string COMMENT '商品名称',
    `sku_desc` string COMMENT '商品描述',
    `weight` string COMMENT '重量',
    `tm_id` string COMMENT '品牌id',
    `category3_id` string COMMENT '3级分类id',
    `category2_id` string COMMENT '2级分类id',
    `category1_id` string COMMENT '1级分类id',
    `category3_name` string COMMENT '3级分类名称',
    `category2_name` string COMMENT '2级分类名称',
    `category1_name` string COMMENT '1级分类名称',
    `create_time` string COMMENT ''
) COMMENT '商品表'
PARTITIONED BY ( `dt` string)
```

```
stored as  parquet
tblproperties ("parquet.compression"="snappy");
```

创建启动日志基础明细表

```
use hdp_zhuanzhuan_dwd_global;
drop table if exists dwd_log_base_start_inc_1d;
CREATE EXTERNAL TABLE dwd_log_base_start_inc_1d(
    `mid_id` string comment '移动互联网设备id',
    `user_id` string comment '用户id',
    `version_code` string comment '版本code',
    `version_name` string comment '版本名称',
    `lang` string comment '系统语言',
    `source` string comment '渠道',
    `os` string comment '操作系统版本',
    `area` string comment '区域',
    `model` string comment '型号',
    `brand` string comment '品牌',
    `sdk_version` string comment 'sdk版本',
    `gmail` string comment '邮箱',
    `height_width` string comment '',
    `app_time` string comment '客户端日志产生时间',
    `network` string comment '网络类型',
    `lng` string comment '经度',
    `lat` string comment '纬度',
    `event_name` string comment '事件名',
    `event_json` string comment '事件详细信息',
    `server_time` string comment '日志服务端时间')
PARTITIONED BY (`dt` string)
stored as  parquet
tblproperties ("parquet.compression"="snappy");
```

创建事件日志基础明细表

```
use hdp_zhuanzhuan_dwd_global;
drop table if exists dwd_log_base_event_inc_1d;
CREATE EXTERNAL TABLE dwd_log_base_event_inc_1d(
    `mid_id` string comment '移动互联网设备id',
    `user_id` string comment '用户id',
    `version_code` string comment '版本code',
    `version_name` string comment '版本名称',
    `lang` string comment '系统语言',
    `source` string comment '终端',
    `os` string comment '操作系统版本',
    `area` string comment '区域',
    `model` string comment '型号',
    `brand` string comment '品牌',
    `sdk_version` string comment 'sdk版本',
    `gmail` string comment '邮箱',
    `height_width` string comment '',
    `app_time` string comment '客户端时间',
    `network` string comment '网络类型',
    `lng` string comment '经度',
    `lat` string comment '纬度',
    `event_name` string comment '事件名',
    `event_json` string comment '事件详细信息',
    `server_time` string comment '日志服务端时间')
PARTITIONED BY (`dt` string)
stored as  parquet
tblproperties ("parquet.compression"="snappy");
```

5.1.4.2. DWD层数据ETL

在 /home/bigdata/warehouse/dwd/bin 下面创建 etl_dwd_data.sh

```
cd /home/bigdata/warehouse/dwd/bin
vim etl_dwd_data.sh
```

在脚本中填写如下内容

```
#!/bin/bash

# 定义变量方便修改
ods_dbname=hdp_zhuanzhuan_ods_global
dwd_dbname=hdp_zhuanzhuan_dwd_global
hive=/home/bigdata/apps/apache-hive-3.1.2-bin/bin/hive

# 如果是输入的日期按照取输入日期；如果没输入日期取当前时间的前一天
if [ -n $1 ] ;then
```

```
    log_date=$1
else
    log_date=`date -d "-1 day" +%F`
fi

sql="

set hive.exec.dynamic.partition.mode=nonstrict;

insert overwrite table "$dwd_dbname".dwd_mysql_order_info_inc_1d partition(dt)
select * from "$ods_dbname".ods_zz_order_info_inc_1d
where dt='$log_date' and id is not null;

insert overwrite table "$dwd_dbname".dwd_mysql_order_detail_inc_1d partition(dt)
select * from "$ods_dbname".ods_zz_order_detail_inc_1d
where dt='$log_date' and id is not null;

insert overwrite table "$dwd_dbname".dwd_mysql_payment_info_inc_1d partition(dt)
select * from "$ods_dbname".ods_zz_payment_info_inc_1d
where dt='$log_date' and id is not null;

insert overwrite table "$dwd_dbname".dwd_mysql_user_info_full_1d partition(dt)
select * from "$ods_dbname".ods_zz_user_info_full_1d
where dt='$log_date' and id is not null;

insert overwrite table "$dwd_dbname".dwd_mysql_sku_info_full_1d partition(dt)
select
    sku.id,
    sku.spu_id,
    sku.price,
    sku.sku_name,
    sku.sku_desc,
    sku.weight,
    sku.tm_id,
    sku.category3_id,
    c2.id category2_id ,
    c1.id category1_id,
    c3.name category3_name,
    c2.name category2_name,
    c1.name category1_name,
    sku.create_time,
    sku.dt
from
"$ods_dbname".ods_zz_sku_info_full_1d sku
join "$ods_dbname".ods_zz_category3_full_1d c3 on sku.category3_id=c3.id
join "$ods_dbname".ods_zz_category2_full_1d c2 on c3.category2_id=c2.id
join "$ods_dbname".ods_zz_category1_full_1d c1 on c2.category1_id=c1.id
where sku.dt='$log_date' and c2.dt='$log_date'
and c3.dt='$log_date' and c1.dt='$log_date'
and sku.id is not null;

insert overwrite table "$dwd_dbname".dwd_log_base_start_inc_1d PARTITION (dt)
select
    mid_id,
    user_id,
    version_code,
    version_name,
    lang,
    source ,
    os ,
    area ,
    model ,
    brand ,
    sdk_version ,
    gmail ,
    height_width ,
    app_time ,
    network ,
    lng ,
    lat ,
    event_name ,
    event_json ,
    server_time ,
    dt
from "$ods_dbname".ods_log_start_app_inc_1d lateral view
json_tuple(line,'mid','uid','vc','vn','l','sr','os','ar','md','ba','sv','g','hw','t','nw','ln','la','en','kv','ett')
tmp_k as
mid_id,user_id,version_code,version_name,lang,source,os,area,model,brand,sdk_version,gmail,height_width,app_time,network
,lng,lat,event_name, event_json,server_time
where dt='$log_date' and event_name <> '' and server_time > 0 and mid_id <> '';

insert overwrite table "$dwd_dbname".dwd_log_base_event_inc_1d PARTITION (dt)
select
```

```
    mid_id,
    user_id,
    version_code,
    version_name,
    lang,
    source ,
    os ,
    area ,
    model ,
    brand ,
    sdk_version ,
    gmail ,
    height_width ,
    app_time ,
    network ,
    lng ,
    lat ,
    event_name ,
    event_json ,
    server_time ,
    dt
from "$ods_dbname".ods_log_event_inc_1d lateral view
json_tuple(line,'mid','uid','vc','vn','l','sr','os','ar','md','ba','sv','g','hw','t','nw','ln','la','en','kv','ett')
tmp_k as
mid_id,user_id,version_code,version_name,lang,source,os,area,model,brand,sdk_version,gmail,height_width,app_time,network
,lng,lat,event_name, event_json,server_time where dt='$log_date' and event_name <> '' and server_time > 0 and mid_id <>
'';
"
$hive -e "$sql"
```

添加脚本执行权限

```
chmod 777 /home/bigdata/warehouse/dwd/bin/etl_dwd_data.sh
```

执行脚本清洗数据

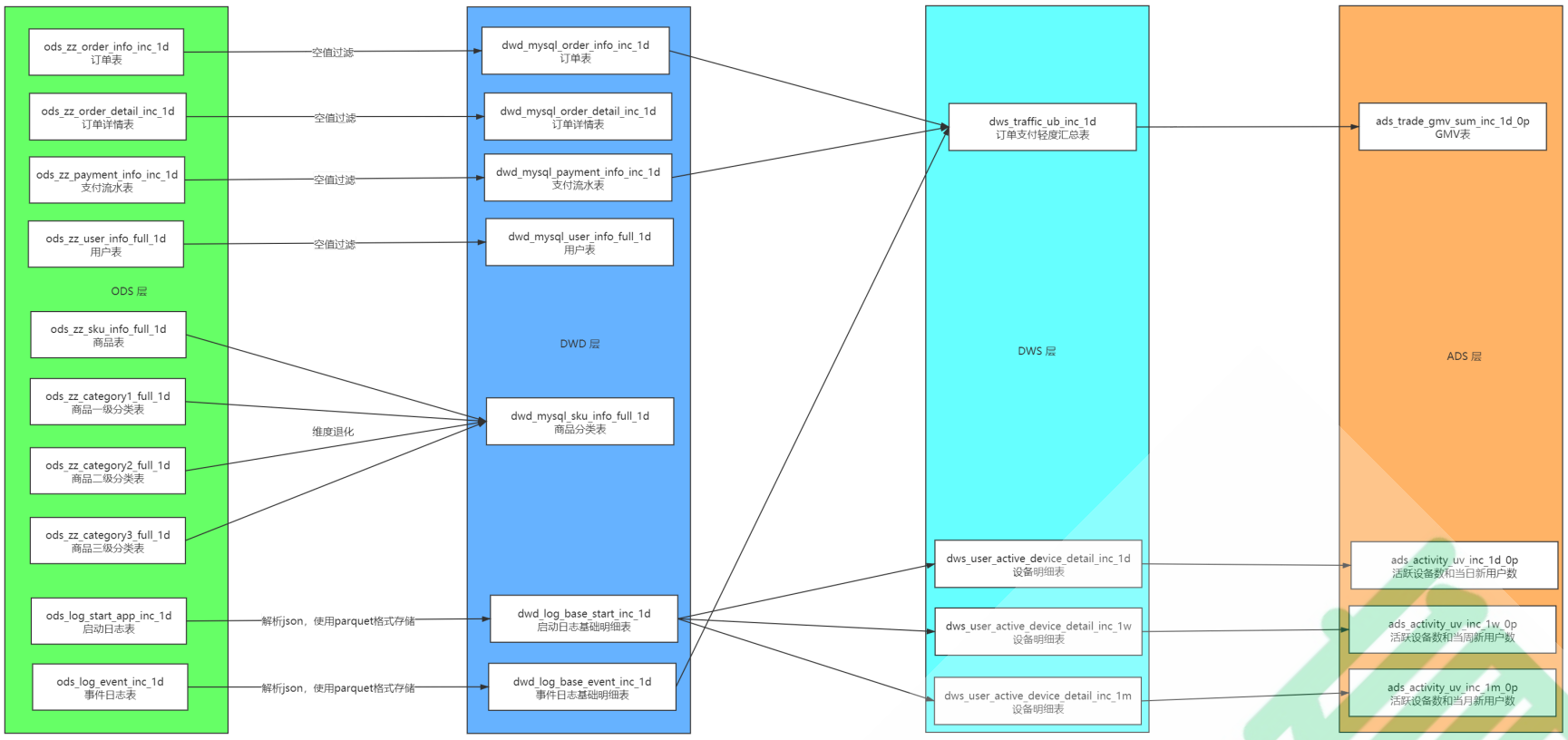
```
cd /home/bigdata/warehouse/dwd/bin
sh etl_dwd_data.sh 2020-09-18
sh etl_dwd_data.sh 2020-09-19
```

检查数据：

```
select * from dwd_log_base_event_inc_1d limit 3;
select * from dwd_log_base_start_inc_1d limit 3;
select * from dwd_mysql_order_detail_inc_1d limit 3;
select * from dwd_mysql_order_info_inc_1d limit 3;
select * from dwd_mysql_payment_info_inc_1d limit 3;
select * from dwd_mysql_sku_info_full_1d limit 3;
select * from dwd_mysql_user_info_full_1d limit 3;
```

记住：从 ODS 到 DWD 做了什么事情：

```
1、订单表，几乎不变，做空值判断
2、支付流水表，几乎不变，做空值判断
3、订单明细，几乎不变，做空值判断
4、商品表：ODS层的四张表做了join处理（维度退化）：  一级分类，二级分类，三级分类，订单明细
5、用户表，也几乎没变，做空值判断
6、启动日志：解析json，使用parquet格式存储
7、事件日志：解析json，使用parquet格式存储
```



5.1.5. DWS层用户行为宽表

为什么要建宽表?

需求目标，把每个用户单日的行为聚合起来组成一张多列宽表，以便之后关联用户维度信息后进行，不同角度的统计分析。

```
我们期望，这张表甚至包含所有的信息，但是明细宽表的构建是违反范式的，利用数据的冗余来提高查询分析的效率！
1、商品表 join 三张 商品分类表
2、先提前去构建这张明细宽表：做任何的分析，都是基于这张宽表，提高查询分析的效率
```

构建非常详细的订单明细宽表：订单信息，用户信息，商品信息，支付信息

行业最佳使用体验：一般来说，可以通过一些 比如 Spark Flink 等进行数据 ETL，当构建出来了大宽表之后，就使用 Clickhouse Doris 等技术针对这张大宽表进提供在线查询分析服务。

5.1.5.1. 创建用户行为宽表

订单支付轻度汇总表/每日用户行为宽表：

```
use hdp_zhuanzhuan_dws_global;
drop table if exists dws_traffic_ub_inc_1d;
create external table dws_traffic_ub_inc_1d
(
    user_id         string      comment '用户 id',
    order_count     bigint      comment '下单次数',
    order_amount    decimal(16,2)  comment '下单金额',
    payment_count   bigint      comment '支付次数',
    payment_amount  decimal(16,2) comment '支付金额',
    comment_count   bigint      comment '评论次数'
) COMMENT '每日用户行为宽表'
PARTITIONED BY (`dt` string)
stored as parquet
tblproperties ("parquet.compression"="snappy");
```

5.1.5.2. 生成每日数据

在 /home/bigdata/warehouse/dws/bin 下面创建 etl_dws_data.sh

```
cd /home/bigdata/warehouse/dws/bin
vim etl_dws_data.sh
```

在脚本中填写如下内容

```
#!/bin/bash

# 定义变量方便修改
dwd_dbname=hdp_zhuanzhuan_dwd_global
dws_dbname=hdp_zhuanzhuan_dws_global
hive=/home/bigdata/apps/apache-hive-3.1.2-bin/bin/hive

# 如果是输入的日期按照取输入日期；如果没输入日期取当前时间的前一天
if [ -n $1 ] ;then
    log_date=$1
```

```
else
    log_date=`date -d "-1 day" +%F`
fi

sql="

with
tmp_order as
(
    select
      user_id,
      sum(oc.total_amount) order_amount,
      count(*)  order_count
    from "$dwd_dbname".dwd_mysql_order_info_inc_1d  oc
    where date_format(oc.create_time,'yyyy-MM-dd')='$log_date'
    group by user_id
),
tmp_payment as
(
    select
      user_id,
      sum(pi.total_amount) payment_amount,
      count(*) payment_count
    from "$dwd_dbname".dwd_mysql_payment_info_inc_1d pi
    where date_format(pi.payment_time,'yyyy-MM-dd')='$log_date'
    group by user_id
),
tmp_comment as
(
    select
      user_id,
      count(*) comment_count
    from "$dwd_dbname".dwd_log_base_event_inc_1d c
    where date_format(c.dt,'yyyy-MM-dd')='$log_date' and c.event_name='comment'
    group by user_id
)
insert overwrite table "$dws_dbname".dws_traffic_ub_inc_1d partition(dt='$log_date')
select
    user_actions.user_id,
    sum(user_actions.order_count),
    sum(user_actions.order_amount),
    sum(user_actions.payment_count),
    sum(user_actions.payment_amount),
    sum(user_actions.comment_count)
from
(
    select
      user_id,
      order_count,
      order_amount ,
      0 payment_count ,
      0 payment_amount,
      0 comment_count
    from tmp_order

    union all
    select
      user_id,
      0,
      0,
      payment_count,
      payment_amount,
      0
    from tmp_payment

    union all
    select
      user_id,
      0,
      0,
      0,
      0,
      comment_count
    from tmp_comment
 ) user_actions
group by user_id;
"

$hive -e "$sql"
```

添加脚本执行权限

```
chmod 777 /home/bigdata/warehouse/dws/bin/etl_dws_data.sh
```

执行脚本清洗数据

```
cd /home/bigdata/warehouse/dws/bin
sh etl_dws_data.sh 2020-09-18
sh etl_dws_data.sh 2020-09-19
```

验证数据：

```
select * from dws_traffic_ub_inc_1d limit 10;
```

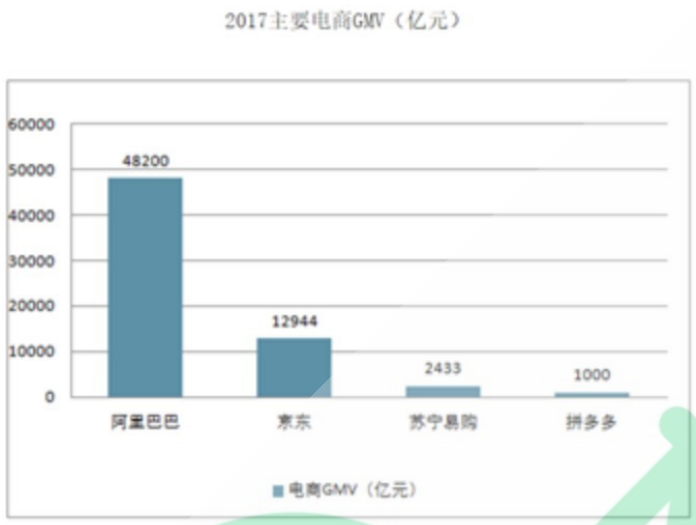
```
use hdp_zhuanzhuan_dws_global;
show partitions dws_traffic_ub_inc_1d;
```

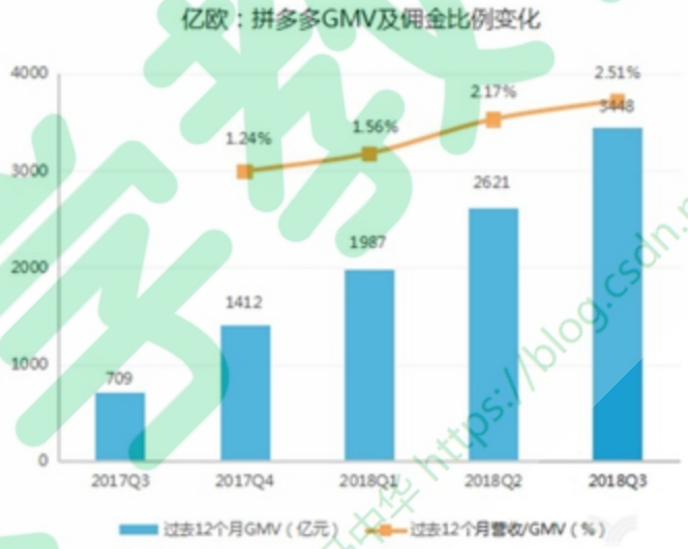
5.2. ADS层数据统计分析

5.2.1. 需求一：GMV成交额分析

GMV：Gross Merchandise Volume，是成交总额（一定时间段内）的意思。

在电商网站定义里面是网站成交金额。这个实际指的是拍下订单金额，包含付款和未付款的部分。GMV是电商平台非常重视的统计指标，甚至写在招股书里。





不同公司计算GMV的算法不同：

GMV＝下单金额

GMV=下单金额 -（大额订单） 10万

GMV=下单金额＋预定金额

日期	当日下单uv	当日支付uv	当日gmv订单数	当日gmv订单额	当日支付订单数	当日支付订单额
2020-09-18	100	50	1000	999999	500	555555

5.2.1.1. 建表：gmv的ads层表

```
use hdp_zhuanzhuan_ads_global;
drop table if exists ads_trade_gmv_sum_inc_1d_0p;
create external table ads_trade_gmv_sum_inc_1d_0p(
    `dt` string COMMENT '统计日期',
    `gmv_uv` bigint comment '当日下单uv',
    `pay_uv` bigint comment '当日支付uv',
    `gmv_pv` bigint  comment '当日gmv订单数',
    `gmv_amount` decimal(16,2) comment '当日gmv下单金额',
    `pay_pv` bigint comment '当日支付订单数',
    `pay_amount` decimal(16,2) comment '当日支付订单额'
) COMMENT '每日gmv汇总统计'
row format delimited  fields terminated by '\t' ;
```

5.2.1.2. 数据统计

SQL实现：

```
set hivevar:stat_date=2020-09-19;
insert into table hdp_zhuanzhuan_ads_global.ads_trade_gmv_sum_inc_1d_0p
select
    '${stat_date}',
    count(distinct case when order_count > 0 then user_id end) gmv_uv,
    count(distinct case when payment_count > 0 then user_id end) pay_uv,
    sum(order_count)  gmv_pv,
    sum(order_amount)  gmv_amount,
    sum(payment_count)  pay_pv,
    sum(payment_amount)  pay_amount
from hdp_zhuanzhuan_dws_global.dws_traffic_ub_inc_1d
where dt ='${stat_date}'
group by dt;
```

5.2.1.3. 查询统计结果

```
select * from hdp_zhuanzhuan_ads_global.ads_trade_gmv_sum_inc_1d_0p where dt='2020-09-18';
```

结果

```
2020-09-18      90491   69207   170889  16559849119.00  107420  2159455438.00
```

5.2.2. 需求二：用户活跃&新增&累计设备主题

5.2.2.1. 相关业务术语

【用户】：用户以设备为判断标准，在移动统计中，每个独立设备认为是一个独立用户。Android 系统根据 IMEI号，IOS 系统根据 OpenUDID 来标识一个独立用户，每部手机一个用户。

【新增用户】：首次联网使用应用的用户。如果一个用户首次打开某 app，那这个用户定义为新增用户；卸载再安装的设备，不会被算作一次新增。新增用户包括日新增用户、周新增用户、月新增用户。

【活跃用户】：打开应用的用户即为活跃用户，不考虑用户的使用情况。每天一台设备打开多次会被计为一个活跃用户。

【周（月）活跃用户】：某个自然周（月）内启动过应用的用户，该周（月）内的多次启动只记一个活跃用户。

【月活跃率】：月活跃用户与截止到该月累计的用户总和之间的比例。

【沉默用户】：用户仅在安装当天（次日）启动一次，后续时间无再启动行为。该指标可以反映新增用户质量和用户与APP的匹配程度。

【版本分布】：不同版本的周内各天新增用户数，活跃用户数和启动次数。利于判断App各个版本之间的优劣和用户行为习惯。

【本周回流用户】：上周未启动过应用，本周启动了应用的用户。

【连续n周活跃用户】：连续n周，每周至少启动一次。

【忠诚用户】：连续活跃5周以上的用户

【连续活跃用户】：连续2周及以上活跃的用户

【近期流失用户】：连续n(2<= n <= 4)周没有启动应用的用户。（第n+1周没有启动过）

【留存用户】：某段时间内的新增用户，经过一段时间后，仍然使用应用的被认作是留存用户；这部分用户占当时新增用户的比例即是留存率。例如，5月份新增用户200，这200人在6月份启动过应用的有100人，7月份启动过应用的有80人，8月份启动过应用的有50人；则5月份新增用户一个月后的留存率是50%，二个月后的留存率是40%，三个月后的留存率是25%。

【用户新鲜度】：每天启动应用的新老用户比例，即新增用户数占活跃用户数的比例。

【单次使用时长】：每次启动使用的时间长度。

【日使用时长】：累计一天内的使用时间长度。

【启动次数计算标准】：IOS 平台应用退到后台就算一次独立的启动；Android 平台我们规定，两次启动之间的间隔小于30秒，被计算一次启动。用户在使用过程中，若因收发短信或接电话等退出应用30秒又再次返回应用中，那这两次行为应该是延续而非独立的，所以可以被算作一次使用行为，即一次启动。业内大多使用30秒这个标准，但用户还是可以自定义此时间间隔。

5.2.2.2. 需求实现

具体需求：统计当日、当周、当月访问的每个设备明细和每日新增设备明细

1、设计创建对应 DWS 层表(每日、每周、每月设备明细表)

```
use hdp_zhuanzhuan_dws_global;
drop table if exists dws_user_active_device_detail_inc_1d;
create external table dws_user_active_device_detail_inc_1d(
    `mid_id` string COMMENT '设备唯一标识',
    `user_id` string COMMENT '用户标识',
    `version_code` string COMMENT '程序版本号',
    `version_name` string COMMENT '程序版本名',
    `lang` string COMMENT '系统语言',
```

```
    `source` string COMMENT '渠道号',
    `os` string COMMENT '安卓系统版本',
    `area` string COMMENT '区域',
    `model` string COMMENT '手机型号',
    `brand` string COMMENT '手机品牌',
    `sdk_version` string COMMENT 'sdkVersion',
    `gmail` string COMMENT 'gmail',
    `height_width` string COMMENT '屏幕宽高',
    `app_time` string COMMENT '客户端日志产生时的时间',
    `network` string COMMENT '网络模式',
    `lng` string COMMENT '经度',
    `lat` string COMMENT '纬度'
) COMMENT '活跃用户按天明细'
PARTITIONED BY ( `dt` string)
stored as  parquet
tblproperties ("parquet.compression"="snappy");
```

```
drop table if exists dws_user_active_device_detail_inc_1w;
create external table dws_user_active_device_detail_inc_1w(
    `mid_id` string COMMENT '设备唯一标识',
    `user_id` string COMMENT '用户标识',
    `version_code` string COMMENT '程序版本号',
    `version_name` string COMMENT '程序版本名',
    `lang` string COMMENT '系统语言',
    `source` string COMMENT '渠道号',
    `os` string COMMENT '安卓系统版本',
    `area` string COMMENT '区域',
    `model` string COMMENT '手机型号',
    `brand` string COMMENT '手机品牌',
    `sdk_version` string COMMENT 'sdkVersion',
    `gmail` string COMMENT 'gmail',
    `height_width` string COMMENT '屏幕宽高',
    `app_time` string COMMENT '客户端日志产生时的时间',
    `network` string COMMENT '网络模式',
    `lng` string COMMENT '经度',
    `lat` string COMMENT '纬度'
) COMMENT '活跃用户按周明细'
PARTITIONED BY (`dt` string)
stored as parquet
tblproperties ("parquet.compression"="snappy");
```

```
drop table if exists dws_user_active_device_detail_inc_1m;
create external table dws_user_active_device_detail_inc_1m(
    `mid_id` string COMMENT '设备唯一标识',
    `user_id` string COMMENT '用户标识',
    `version_code` string COMMENT '程序版本号',
    `version_name` string COMMENT '程序版本名',
    `lang` string COMMENT '系统语言',
    `source` string COMMENT '渠道号',
    `os` string COMMENT '安卓系统版本',
    `area` string COMMENT '区域',
    `model` string COMMENT '手机型号',
    `brand` string COMMENT '手机品牌',
    `sdk_version` string COMMENT 'sdkVersion',
    `gmail` string COMMENT 'gmail',
    `height_width` string COMMENT '屏幕宽高',
    `app_time` string COMMENT '客户端日志产生时的时间',
    `network` string COMMENT '网络模式',
    `lng` string COMMENT '经度',
    `lat` string COMMENT '纬度'
) COMMENT '活跃用户按月明细'
PARTITIONED BY ( `dt` string)
stored as  parquet
tblproperties ("parquet.compression"="snappy");
```

2、数据清洗

按天统计表：

```
set hivevar:stat_date=2020-09-18;
set hive.exec.dynamic.partition.mode=nonstrict;
insert overwrite table hdp_zhuanzhuan_dws_global.dws_user_active_device_detail_inc_1d partition(dt)
select
    mid_id,
    collect_set(user_id)[0] user_id,
    collect_set(version_code)[0] version_code,
    collect_set(version_name)[0] version_name,
    collect_set(lang)[0]lang,
    collect_set(source)[0] source,
    collect_set(os)[0] os,
```

```
    collect_set(area)[0] area,
    collect_set(model)[0] model,
    collect_set(brand)[0] brand,
    collect_set(sdk_version)[0] sdk_version,
    collect_set(gmail)[0] gmail,
    collect_set(height_width)[0] height_width,
    collect_set(app_time)[0] app_time,
    collect_set(network)[0] network,
    collect_set(lng)[0] lng,
    collect_set(lat)[0] lat,
    '${stat_date}'
from hdp_zhuanzhuan_dwd_global.dwd_log_base_start_inc_1d
where dt='${stat_date}'
group by mid_id;
```

按周统计表：

```
set hivevar:stat_date=2020-09-18;
set hive.exec.dynamic.partition.mode=nonstrict;
insert overwrite table hdp_zhuanzhuan_dws_global.dws_user_active_device_detail_inc_1w partition(dt)
select
    mid_id,
    collect_set(user_id)[0] user_id,
    collect_set(version_code)[0] version_code,
    collect_set(version_name)[0] version_name,
    collect_set(lang)[0] lang,
    collect_set(source)[0] source,
    collect_set(os)[0] os,
    collect_set(area)[0] area,
    collect_set(model)[0] model,
    collect_set(brand)[0] brand,
    collect_set(sdk_version)[0] sdk_version,
    collect_set(gmail)[0] gmail,
    collect_set(height_width)[0] height_width,
    collect_set(app_time)[0] app_time,
    collect_set(network)[0] network,
    collect_set(lng)[0] lng,
    collect_set(lat)[0] lat,
    date_add(next_day('${stat_date}','MO'), -1)
from hdp_zhuanzhuan_dwd_global.dwd_log_base_start_inc_1d
where dt>=date_add(next_day('${stat_date}','MO'),-7) and dt<=date_add(next_day('${stat_date}','MO'),-1)
group by mid_id;
```

按月统计表：

```
set hivevar:stat_date=2020-09-18;
set hive.exec.dynamic.partition.mode=nonstrict;
insert overwrite table hdp_zhuanzhuan_dws_global.dws_user_active_device_detail_inc_1m partition(dt)
select
    mid_id,
    collect_set(user_id)[0] user_id,
    collect_set(version_code)[0] version_code,
    collect_set(version_name)[0] version_name,
    collect_set(lang)[0] lang,
    collect_set(source)[0] source,
    collect_set(os)[0] os,
    collect_set(area)[0] area,
    collect_set(model)[0] model,
    collect_set(brand)[0] brand,
    collect_set(sdk_version)[0] sdk_version,
    collect_set(gmail)[0] gmail,
    collect_set(height_width)[0] height_width,
    collect_set(app_time)[0] app_time,
    collect_set(network)[0] network,
    collect_set(lng)[0] lng,
    collect_set(lat)[0] lat,
    date_format('${stat_date}','yyyy-MM')
from hdp_zhuanzhuan_dwd_global.dwd_log_base_start_inc_1d
where date_format(dt,'yyyy-MM') = date_format('${stat_date}','yyyy-MM')
group by mid_id;
```

3、设计创建对应DWS层表(每日新增设备明细表)

```
use hdp_zhuanzhuan_dws_global;
drop table if exists dws_user_new_device_detail_inc_1d;
create external table dws_user_new_device_detail_inc_1d(
    `mid_id` string COMMENT '设备唯一标识',
    `user_id` string COMMENT '用户标识',
    `version_code` string COMMENT '程序版本号',
    `version_name` string COMMENT '程序版本名',
```

```
    `lang` string COMMENT '系统语言',
    `source` string COMMENT '渠道号',
    `os` string COMMENT '安卓系统版本',
    `area` string COMMENT '区域',
    `model` string COMMENT '手机型号',
    `brand` string COMMENT '手机品牌',
    `sdk_version` string COMMENT 'sdkVersion',
    `gmail` string COMMENT 'gmail',
    `height_width` string COMMENT '屏幕宽高',
    `app_time` string COMMENT '客户端日志产生时的时间',
    `network` string COMMENT '网络模式',
    `lng` string COMMENT '经度',
    `lat` string COMMENT '纬度'
) COMMENT '新用户设备按天明细'
PARTITIONED BY ( `dt` string)
stored as  parquet
tblproperties ("parquet.compression"="snappy");
```

4、数据清洗

```
set hivevar:stat_date=2020-09-18;
set hive.exec.dynamic.partition.mode=nonstrict;
insert overwrite table dws_user_new_device_detail_inc_1d partition (dt)
select
dau.mid_id,
dau.user_id ,
dau.version_code ,
dau.version_name ,
dau.lang ,
dau.source,
dau.os,
dau.area,
dau.model,
dau.brand,
dau.sdk_version,
dau.gmail,
dau.height_width,
dau.app_time,
dau.network,
dau.lng,
dau.lat,
'${stat_date}'
from hdp_zhuanzhuan_dws_global.dws_user_active_device_detail_inc_1d dau left join
hdp_zhuanzhuan_dws_global.dws_user_new_device_detail_inc_1d nd on dau.mid_id=nd.mid_id and nd.dt<'${stat_date}'
where dau.dt='${stat_date}' and nd.mid_id is null;
```

查询结果数据：

```
select * from hdp_zhuanzhuan_dws_global.dws_user_new_device_detail_inc_1d limit 10;
```

5、ADS层计算当日、当周、当月活跃设备数和当日新用户数

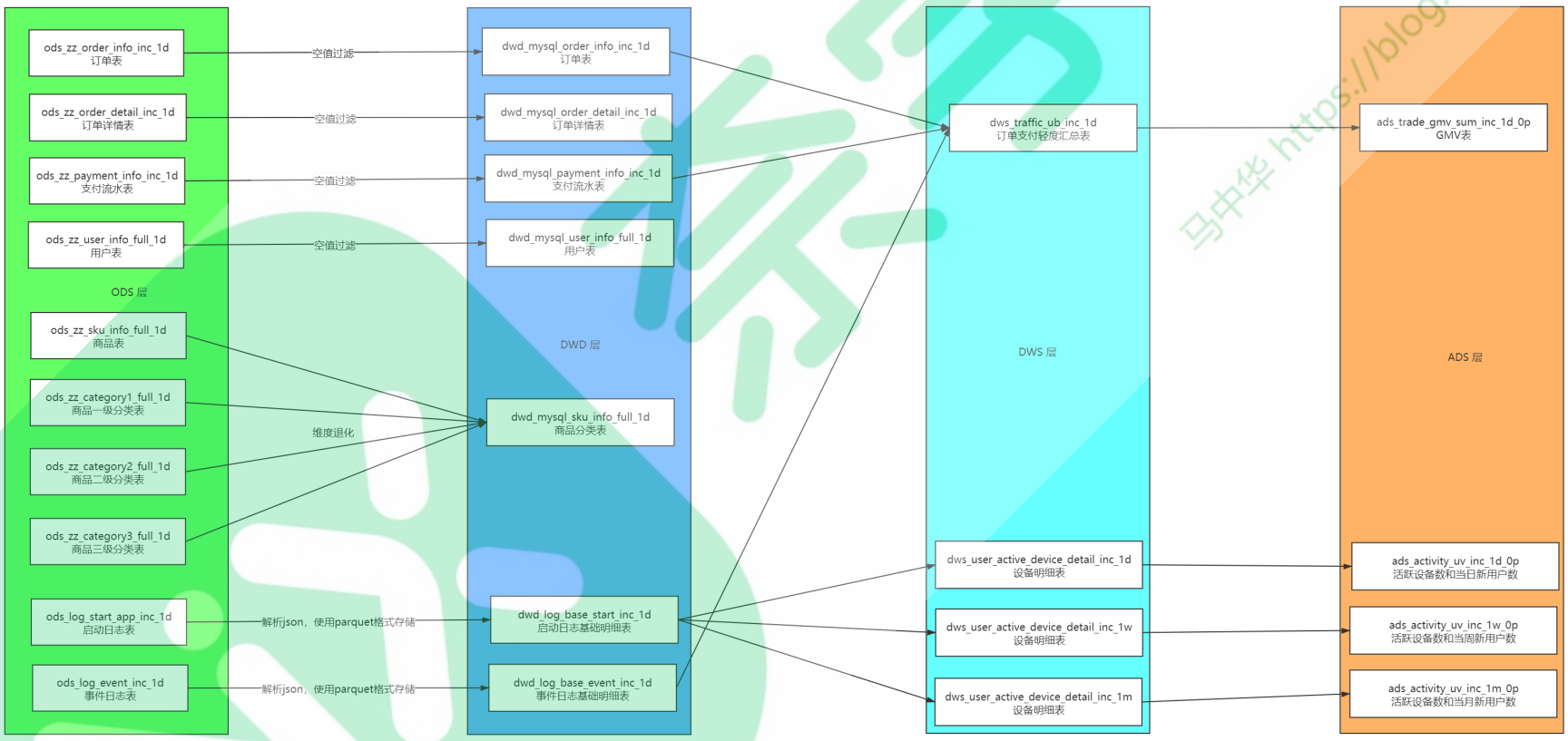
```
use hdp_zhuanzhuan_ads_global;
drop table if exists ads_activity_uv_inc_1d_0p;
create external table ads_activity_uv_inc_1d_0p
(
    `dt` string COMMENT '统计日期',
    `dau_count` bigint COMMENT '当日用户数量',
    `wau_count`  bigint COMMENT '当周用户数量',
    `mau_count`  bigint COMMENT '当月用户数量',
    `nau_count`  bigint COMMENT '当日新用户数量',
    `is_weekend` string COMMENT 'Y,N是否是周末,用于得到本周最终结果',
    `is_monthend` string COMMENT 'Y,N是否是月末,用于得到本月最终结果'
) COMMENT '每日活跃用户汇总统计'
row format delimited  fields terminated by '\t' ;
```

```
set hivevar:stat_date=2020-09-18;
insert into table hdp_zhuanzhuan_ads_global.ads_activity_uv_inc_1d_0p
select
  '${stat_date}' dt,
   d.ct,
   w.ct,
   m.ct,
   n.ct,
   if(date_add(next_day('${stat_date}','MO'), -1)='${stat_date}', 'Y', 'N') ,
   if(last_day('${stat_date}')='${stat_date}', 'Y', 'N')
from
```

```
(
    select
    '${stat_date}' dt,
    count(mid_id) ct
    from hdp_zhuanzhuan_dws_global.dws_user_active_device_detail_inc_1d
    where dt='${stat_date}'
)d join
(
    select
    '${stat_date}' dt,
    count (mid_id) ct
    from hdp_zhuanzhuan_dws_global.dws_user_active_device_detail_inc_1w
    where dt=date_add(next_day('${stat_date}','MO'), -1)
)w on d.dt=w.dt
join
(
    select
    '${stat_date}' dt,
    count (mid_id) ct
    from hdp_zhuanzhuan_dws_global.dws_user_active_device_detail_inc_1m
    where dt=date_format('${stat_date}','yyyy-MM')
)m on d.dt=m.dt
join
(
    select
    '${stat_date}' dt,
    count (mid_id) ct
    from hdp_zhuanzhuan_dws_global.dws_user_new_device_detail_inc_1d
    where dt='${stat_date}'
)n on d.dt=n.dt;
```

查询结果数据：

```
select * from hdp_zhuanzhuan_ads_global.ads_activity_uv_inc_1d_0p limit 10;
```



5.2.3. 需求三：用户留存主题

5.2.3.1. 用户留存概念

留存用户：某段时间内的新增用户（活跃用户），经过一段时间后，又继续使用应用的被认作是留存用户；

留存率：留存用户占当时新增用户（活跃用户）的比例即是留存率。

例如，2月10日新增用户100，这100人在2月11日启动过应用的有30人，2月12日启动过应用的有25人，2月13日启动过应用的有32人；

则2月10日新增用户次日的留存率是30/100 = 30%，两日留存率是25/100=25%，三日留存率是32/100=32%。

时间	新增用户	1天后	2天后	3天后
2019-02-10	100	30% (2-11)	25% (2-12)	32% (2-13)
2019-02-11	200	20% (2-12)	15% (2-13)	
2019-02-12	100	25% (2-13)		
2019-02-13				

5.2.3.2. 统计每天1、3、7、30日留存率

1、DWS层（每日留存用户明细表）创建

```
use hdp_zhuanzhuan_dws_global;
drop table if exists dws_user_new_device_retention_inc_1d;
create external table dws_user_new_device_retention_inc_1d(
    `mid_id` string COMMENT '设备唯一标识',
    `user_id` string COMMENT '用户标识',
    `version_code` string COMMENT '程序版本号',
    `version_name` string COMMENT '程序版本名',
    `lang` string COMMENT '系统语言',
    `source` string COMMENT '渠道号',
    `os` string COMMENT '安卓系统版本',
    `area` string COMMENT '区域',
    `model` string COMMENT '手机型号',
    `brand` string COMMENT '手机品牌',
    `sdk_version` string COMMENT 'sdkVersion',
    `gmail` string COMMENT 'gmail',
    `height_width` string COMMENT '屏幕宽高',
    `app_time` string COMMENT '客户端日志产生时的时间',
    `network` string COMMENT '网络模式',
    `lng` string COMMENT '经度',
    `lat` string COMMENT '纬度',
    `create_date`  string  comment '设备新增时间',
    `retention_day`  int comment '截止当前日期留存天数'
) COMMENT '每日新用户留存明细'
PARTITIONED BY ( `dt` string)
stored as  parquet
tblproperties ("parquet.compression"="snappy");
```

2、清洗数据（每天计算前1,3,7,n天的新用户访问留存明细）

```
set hivevar:stat_date=2020-09-18;
insert overwrite table hdp_zhuanzhuan_dws_global.dws_user_new_device_retention_inc_1d partition(dt='${stat_date}')
select
    nm.mid_id,
    nm.user_id ,
    nm.version_code ,
    nm.version_name ,
    nm.lang ,
    nm.source,
    nm.os,
    nm.area,
    nm.model,
    nm.brand,
    nm.sdk_version,
    nm.gmail,
    nm.height_width,
    nm.app_time,
    nm.network,
    nm.lng,
    nm.lat,
    nm.dt,
    datediff(ud.dt,nm.dt) retention_day
from hdp_zhuanzhuan_dws_global.dws_user_active_device_detail_inc_1d ud join
hdp_zhuanzhuan_dws_global.dws_user_new_device_detail_inc_1d nm on ud.mid_id = nm.mid_id
```

```
where ud.dt='${stat_date}' and nm.dt in(date_add('${stat_date}',-1), date_add('${stat_date}',-3),
date_add('${stat_date}',-7),date_add('${stat_date}',-30));
```

注意上述 SQL 的两个细节：

```
1、nm.dt in(date_add('${stat_date}',-1), date_add('${stat_date}',-3),
date_add('${stat_date}',-7),date_add('${stat_date}',-30));
计算得到今天的所有用户中，有多少是在一日以前注册的，有多少是在3日以前注册的
2、ud.mid_id = nm.mid_id 根据新增设备明细表做链接
```

结果集的样子：

```
今天：2020-09-18
    1：2020-09-17
    3：2020-09-15
    7：2020-09-11
    30：2020-08-18

日期          注册     一日   三日     7日    30日
2020-09-18    1000    200   180      30    150
2020-09-19    2000    200   180      30    150
```

思路分两步走：

```
1、先求得过去1天，过去3天，过去7天，过去30这个四个日期的新增用户  联合到一起形成一个集合
    四个值（四个日期的新增用户） + 一个集合（四个日期的新增用户合起来作为条件作为今天的筛选条件）
2、在这个集合中，还有多少用户是今天登陆使用了APP的 （200个）
    （result 交集 过去1天） / 过去1天
```

3、查询数据（每天计算前1,3,7,30天的新用户访问留存明细）

```
set hivevar:stat_date=2020-09-18;
select retention_day, count(*) from hdp_zhuanzhuan_dws_global.dws_user_new_device_retention_inc_1d where
dt='${stat_date}' group by retention_day;
```

4、ADS 层留存用户数

三日留存率： 对于过去3天那一天来说，用几天的登陆用户 / 过去三天那一天的新增用户

三日留存率: 2020-09-18登录用户 / 2020-09-15新注册用户

```
use hdp_zhuanzhuan_ads_global;
drop table if exists ads_activity_new_dau_retention_inc_1d_0p;
create external table ads_activity_new_dau_retention_inc_1d_0p
(
    `dt` string COMMENT '设备新增日期',
    `retention_day` int comment '截止当前日期留存天数',
    `retention_count` bigint comment  '留存数量'
) COMMENT '每日用户留存统计'
row format delimited  fields terminated by '\t';
```

```
set hivevar:stat_date=2020-09-18;
insert into table hdp_zhuanzhuan_ads_global.ads_activity_new_dau_retention_inc_1d_0p
select
    create_date,
    retention_day,
    count(*) retention_count
from hdp_zhuanzhuan_dws_global.dws_user_new_device_retention_inc_1d
where dt='${stat_date}'
group by create_date,retention_day;
```

5.3. 订单拉链表设计

5.3.1. 什么是拉链表

拉链表，**记录每条信息的生命周期为单位**，一条记录的生命周期结束，就重新开始一条新的记录，并把当前日期放入生效开始日期。

如果当前信息至今有效，在生效结束日期中植入一个极大值（如 9999-99-99 或者 9999-12-31）

订单ID	订单额	状态	生效开始日期	生效结束日期
1	2000.00	待支付	2020-01-01	9999-99-99
2	**1000.00**	**待支付**	**2020-01-01**	**2020-01-01**
2	**1000.00**	**已支付**	**2020-01-02**	**9999-99-99**
3	210.00	已支付	2020-01-01	9999-99-99
4	3300.00	待支付	2020-01-02	9999-99-99
5	55.00	已支付	2020-01-02	9999-99-99

5.3.2. 为什么要做拉链表

1）需要查看某些业务信息的某个时间点当日的信息

2）数据会发生变化，但是大部分是不变的。比如订单信息从下单、支付、发货、签收等状态经历了一周，大部分时间是不变的。（无法每日做增量）

3）数据量有一定规模，无法按照每日全量的方式保存。比如：10亿订单*365天，每天一份订单

5.3.3. 拉链表形成过程

1月2号的拉链表 = 1月1号的拉链表 union 1月2号的变化表

1）假设， 2019年 1月1日的订单全量表是最初始的订单表，如下

订单 Id	状态
1	待支付
2	待支付
3	已支付

2）初始的拉链表就等于最开始的2019年 1月1日的订单全量表

订单 Id	状态	生效开始日期	生效结束日期
1	待支付	2019-01-01	9999-99-99
2	待支付	2019-01-01	9999-99-99
3	已支付	2019-01-01	9999-99-99

3）第二天 1月2日 订单全量表（订单2发生状态修改；订单4、5增加）

订单 Id	状态
1	待支付
2	已支付
3	已支付
4	待支付
5	已支付

4）根据订单表的创建时间和操作时间，得到订单变化表。

订单 Id	状态
2	已支付
4	待支付
5	已支付

5）订单变化表与之前的拉链表合并得到

订单 Id	状态	生效开始日期	生效结束日期
1	待支付	2019-01-01	9999-99-99
2	待支付	2019-01-01	2019-01-01
2	已支付	2019-01-02	9999-99-99
3	已支付	2019-01-01	9999-99-99
4	待支付	2019-01-02	9999-99-99
5	已支付	2019-01-02	9999-99-99

通过，某个日期<=生效开始日期 且 某个日期>=生效结束日期，能够得到某个时间点的数据全量切片。

1）拉链表数据

订单 Id	状态	生效开始日期	生效结束日期
1	待支付	2019-01-01	9999-99-99
2	待支付	2019-01-01	2019-01-01
2	已支付	2019-01-02	9999-99-99
3	已支付	2019-01-01	9999-99-99
4	待支付	2019-01-02	9999-99-99
5	已支付	2019-01-02	9999-99-99

2）例如获取2019-01-01的历史切片：select * from x_order where start_date<='2019-01-01' and end_date>='2019-01-01'

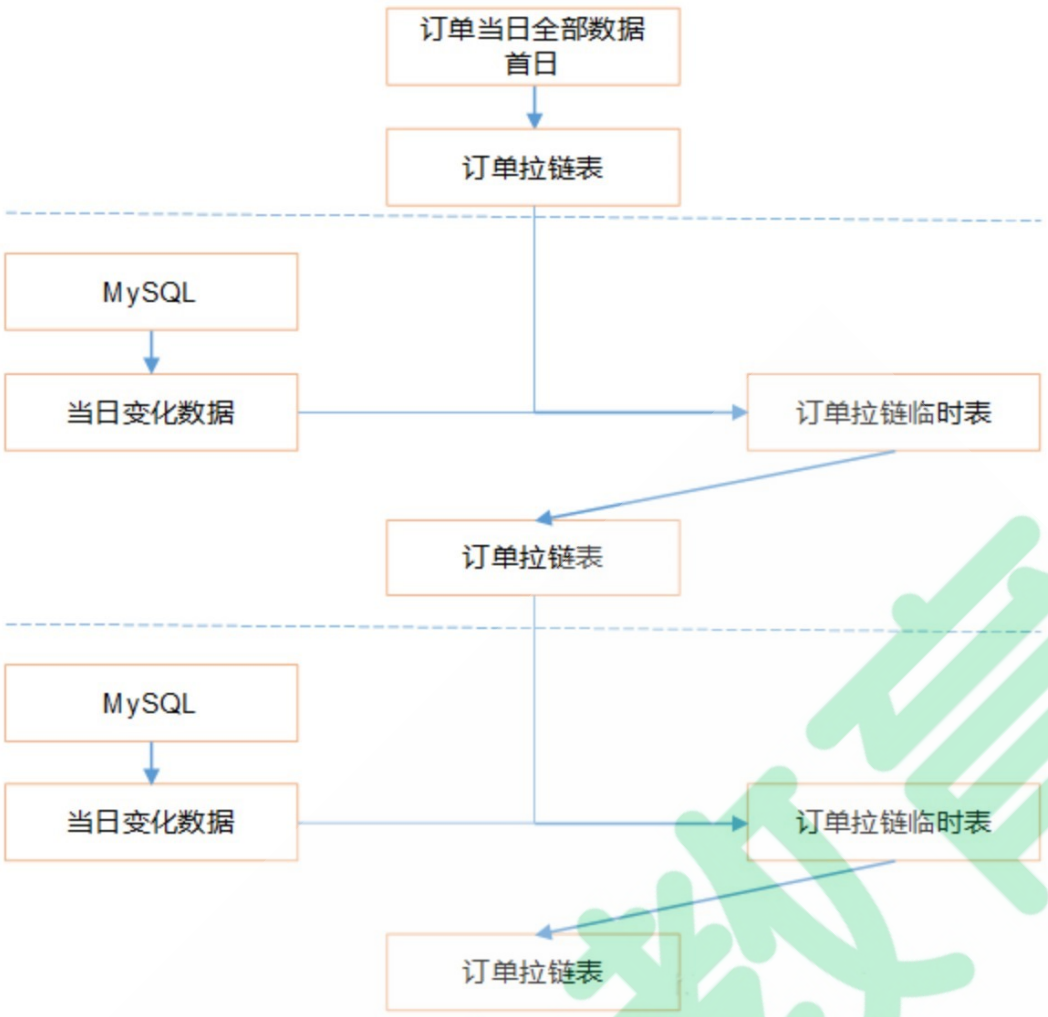
订单 Id	状态	生效开始日期	生效结束日期
1	待支付	2019-01-01	9999-99-99
2	待支付	2019-01-01	2019-01-01
3	已支付	2019-01-01	9999-99-99

3）例如获取2019-01-02的历史切片： select * from x_order where start_date<='2019-01-02' and end_date>='2019-01-02'

订单 Id	状态	生效开始日期	生效结束日期
1	待支付	2019-01-01	9999-99-99
2	已支付	2019-01-02	9999-99-99
3	已支付	2019-01-01	9999-99-99
4	待支付	2019-01-02	9999-99-99
5	已支付	2019-01-02	9999-99-99

5.3.4. 拉链表制作过程

订单当日全部数据和MySQL中每天变化的数据拼接在一起，形成一个新的临时拉链表数据。用临时的拉链表覆盖旧的拉链表数据。（这就解决了hive表中数据不能更新的问题）



1、初始化拉链表，首次独立执行

```
use hdp_zhuanzhuan_dwd_global;
drop table if exists dwd_mysql_order_his_full_1d_0p;
create external table dwd_mysql_order_his_full_1d_0p (
    `id` string COMMENT '订单编号',
    `total_amount` decimal(10,2) COMMENT '订单金额',
    `order_status` string COMMENT '订单状态',
    `user_id` string COMMENT '用户id',
    `payment_way` string COMMENT '支付方式',
    `out_trade_no` string COMMENT '支付流水号',
    `create_time` string COMMENT '创建时间',
    `operate_time` string COMMENT '操作时间',
    `start_date`  string COMMENT '有效开始日期',
    `end_date`  string COMMENT '有效结束日期'
) COMMENT '订单拉链表'
stored as parquet
tblproperties ("parquet.compression"="snappy");
```

```
set hivevar:stat_date=2020-09-18;
insert overwrite table hdp_zhuanzhuan_dwd_global.dwd_mysql_order_his_full_1d_0p
select
    id,
    total_amount,
    order_status,
    user_id,
    payment_way,
    out_trade_no,
    create_time,
    operate_time,
    '${stat_date}',
    '9999-99-99'
from hdp_zhuanzhuan_ods_global.ods_zz_order_info_inc_1d oi
where oi.dt<='${stat_date}';
```

2、获取当日变动（包括新增，修改）每日执行

如何获得每日变动表

（a）最好表内有创建时间和变动时间

（b）如果没有，可以利用第三方工具监控比如 canal，监控 MySQL 的实时变化进行记录(麻烦)。

（c）逐行对比前后两天的数据, 检查 md5(concat(全部有可能变化的字段))是否相同(low)

（d）要求业务数据库提供变动流水(人品，颜值)

因为 dwd_mysql_order_info_inc_1d 本身导入过来就是新增变动明细的表，所以不用处理

3、先合并变动信息，再追加新增信息覆盖更新 dwd_mysql_order_his_full_1d_0p

```
set hivevar:stat_date=2020-09-19;
drop table if exists hdp_zhuanzhuan_tmp_global.tmp_dwd_mysql_order_his_full;
create table hdp_zhuanzhuan_tmp_global.tmp_dwd_mysql_order_his_full as
select * from
(
    select id ,
    total_amount ,
    order_status ,
    user_id ,
    payment_way ,
    out_trade_no ,
    create_time ,
    operate_time ,
    '${stat_date}' start_date,
    '9999-99-99' end_date
    from hdp_zhuanzhuan_dwd_global.dwd_mysql_order_info_inc_1d where dt='${stat_date}'

    union all

    select oh.id,
    oh.total_amount,
    oh.order_status,
    oh.user_id,
    oh.payment_way,
    oh.out_trade_no,
    oh.create_time,
    oh.operate_time,
    oh.start_date,
    if(oi.id is null, oh.end_date, date_add(oi.dt, -1)) end_date
    from hdp_zhuanzhuan_dwd_global.dwd_mysql_order_his_full_1d_0p oh left join
    (
        select
        *
        from hdp_zhuanzhuan_dwd_global.dwd_mysql_order_info_inc_1d
        where dt = '${stat_date}'
    ) oi
    on oh.id = oi.id and oh.end_date = '9999-99-99'
) his;
insert overwrite table hdp_zhuanzhuan_dwd_global.dwd_mysql_order_his_full_1d_0p
select * from hdp_zhuanzhuan_tmp_global.tmp_dwd_mysql_order_his_full;
```

4、获取某个时间的全量切片数据

```
set hivevar:stat_date=2020-09-19;
select t.* from hdp_zhuanzhuan_dwd_global.dwd_mysql_order_his_full_1d_0p t where start_date<='${stat_date}' and
end_date>='${star_date}' limit 10;
```

6. 本次课程总结

今天的主要内容是：

```
1、数仓搭建ODS、DWD与DWS层
2、ADS层数据统计分析
3、订单拉链表设计
```

7. 课程结束，删除所有数据

删除有库：

```
drop database hdp_zhuanzhuan_ods_global cascade;
drop database hdp_zhuanzhuan_dwd_global cascade;
drop database hdp_zhuanzhuan_dws_global cascade;
drop database hdp_zhuanzhuan_ads_global cascade;
drop database hdp_zhuanzhuan_dim_global cascade;
drop database hdp_zhuanzhuan_tmp_global cascade;
```

8. 实时数仓

1、Flink SparkStreaming 实时 ETL （ODS, DWD, DWS 等都在 Flink 中完成，生成的 基础明细数据大宽表）

2、构建大宽表实时入库 ClickHouse Doris